SQL Intro

References

- [RG] Sec 3.1-3.4, 5.1-5.2

Admin info

- Please make sure you know how to find CMS and Piazza
  - cms.csuglab.cornell.edu
  - piazza.com
- Quiz on course policies due 13th Feb

Relational model

- Mathematical abstraction
- Abstraction provided by a database such as MySQL
- Closely related, used interchangeably, but technically not the same
  - Differences?
    - Duplicates
    - Ordering

Querying Relational Data

- Several query abstractions/languagse
- **SQL** (Structured Query Language)
- **Relational Algebra**
  - Used as intermediate representation in your SQL engine/DBMS

Creating Relations in SQL

- Creates Students relation
  - Type (domain) of each field is specified
  - Enforced by DBMS whenever tuples are added or modified
- Enrolled table holds information about courses that students take

```
CREATE TABLE Students
        (sid CHAR(20),
         name CHAR(20),
         login CHAR(10),
         age INT,
         gpa REAL);

CREATE TABLE Enrolled
        (sid CHAR(20),
         cid CHAR(20),
         grade CHAR(2));
```

Destroying and Altering Relations

```
DROP TABLE  Students;
```

Destroys the relation Students
- Schema information and tuples are deleted

```
ALTER TABLE  Students
        ADD COLUMN nationality  CHAR(30);
```

Schema of Students is altered by adding a new field

- Every tuple in current instance is extended with a null value in the new field
- Type DESCRIBE Students to see schema

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students
VALUES (12345, 'Smith', 'smith@ee', 18, 3.2, 'US');
```

- Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE FROM Students
WHERE name = 'Smith';
```

Integrity Constraints (ICs)

- Conditions that must be true for any instance of the database
- Specified when schema is defined
  - or with ALTER statement
- Enforced by the DBMS

Primary Key Constraints

- A set of fields is a key for a relation if:
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key
     - Part 2 false? A superkey
- If there is >1 key for a relation, one of the keys is chosen (by DBA) to be the primary key

ALTER TABLE STUDENTS ADD PRIMARY KEY (SID);

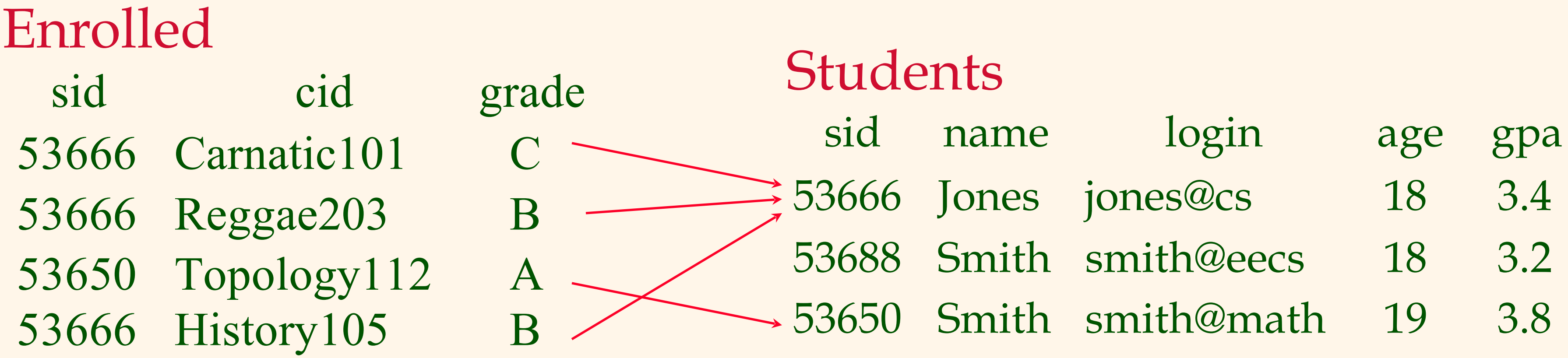
Foreign Keys

- Foreign key: set of fields in one relation that is used to "refer" to a tuple in another relation
  - Usually must correspond to primary key of second relation
- Enrolled(sid: string, cid: string, grade: string)
  - Any sid in Enrolled must also be in Students
    - sid in Enrolled is a foreign key referencing sid in students
- If all foreign key constraints are enforced, referential integrity is achieved

Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses

```
CREATE TABLE Enrolled
  (sid CHAR(20),  cid CHAR(20),  grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid) REFERENCES Students (sid) );
```



Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enforcing Referential Integrity

- What should be done if an Enrolled tuple with a non-existent student id is inserted?
  - Reject it
- What should be done if a Students tuple is deleted?
  - Also delete all Enrolled tuples that refer to it
  - Disallow deletion of a Students tuple that is referred to
  - Set sid in Enrolled tuples that refer to it to a default sid.
  - Set sid in Enrolled tuples that refer to it to a special value null, denoting `unknown’ or `inapplicable’
- Similar if primary key of Students tuple is updated

Referential Integrity in SQL

- 4 options on deletes and updates
  - Default is NO ACTION (delete/update is rejected)
  - CASCADE (also delete all tuples that refer to deleted tuple)
  - SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
   (sid CHAR(20),
    cid CHAR(20),
    grade CHAR(2),
    PRIMARY KEY  (sid,cid),
    FOREIGN KEY (sid)
      REFERENCES Students
           ON DELETE CASCADE
           ON UPDATE CASCADE);
```

So far

- Relations
- Constraints:
  - Attribute types
  - Keys
  - Foreign Keys
- CREATE, ALTER, INSERT, DELETE

Queries

- Now, how do we get data out of the database?
- Using SQL queries
- Lots of very powerful features

Running example - Sailors and Boats

- Sailors in a club reserve Boats
- Tables demo
- Example data in CMS

Basic SQL Query

```
SELECT    S.sname
FROM      Sailors S
WHERE     S.age > 20;
```

```
SELECT DISTINCT   S.sname
FROM      Sailors S
WHERE     S.age > 20;
```

- Default is that duplicates are *not* eliminated!
  - Need to explicitly say “DISTINCT”

Range variables/aliases

- Not required but considered "good practice"
  - Especially once we get to multi-relation queries
- All three below queries are identical

```
SELECT    S.sname
FROM      Sailors S
WHERE     S.age > 20;
```

```
SELECT    sname
FROM      Sailors
WHERE     age > 20;
```

```
SELECT    Sailors.sname
FROM      Sailors
WHERE     Sailors.age > 20;
```

Basic SQL Query

```
SELECT       [DISTINCT]  target-list
[FROM         relation-list]
[WHERE        condition]
```

- No FROM-clause often not supported
- SELECT * returns all attributes

SQL Query

```
SELECT  S.sname
FROM    Sailors S, Reserves R
WHERE   S.sid=R.sid AND R.bid=101;
```

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of *relation-list*
  - Discard resulting tuples if they fail *condition*.
  - Delete attributes that are not in *target-list*
  - If DISTINCT is specified, eliminate duplicate rows.
- This strategy is probably the least efficient way to compute a query!
  - An optimizer will find more efficient strategies to compute *the same answers*.

JOINs

- A common operation: find all "matching" pairs of tuples from two relations

```
SELECT * FROM SAILORS S, RESERVES R WHERE R.sid = S.sid;
```

- Our example query can be thought of as a two step process:
  - JOIN Sailors and Reserves
  - Retain only tuples where bid = 101;

More JOIN syntax

- Other ways to compute a JOIN between R and S in MySQL and PostgreSQL:

```
SELECT * FROM SAILORS S JOIN RESERVES R ON (S.sid = R.sid);

SELECT * FROM SAILORS JOIN RESERVES USING (sid);

SELECT * FROM SAILORS NATURAL JOIN RESERVES;
```

- But beware of the NATURAL JOIN
  - Make sure you really know what the columns with the same names are!
  - And some systems (notably MS SQL Server) don’t support it

Expressions and Strings

```
SELECT  S.age, S.age > 30 AS isOver30, 2*S.age AS age2
FROM    Sailors S
WHERE   S.sname  LIKE 'B_%B';
```

- *Find triples (of ages of sailors and two fields defined by expressions) for sailors whose names begin and end with B and contain at least three characters.*
- AS is used to name fields in result.
- LIKE is used for string matching
  - `_’ stands for any one character
  - `%’ stands for 0 or more arbitrary characters.